MCP SERVER | NO CODE | CLOUD HOSTED

Forgejo (Gitea Fork) MCP

Manage CI/CD and Repo Access in Chat

Forgejo (Gitea Fork) MCP connects your self-hosted Git instances to any AI agent. It lets you manage repository access, check version compatibility across multiple environments, and trigger CI/CD workflows—all from a single conversation. Need an API token or running a deployment test? This is where you do it.

F Quality Score 43.65/100

git-repository | self-hosted | version-control | ci-cd | token-management | automation

MAINTAINED BY VINKIUS CLOUD FOR AI AGENTS · JUNE 2026
vinkius.com/mcp/forgejo-gitea-fork

SCAN TO VIEW

The infrastructure that powers AI agents in the real world.

Vinkius connects AI to the world's software through secure, enterprise-grade infrastructure — enabling real-world execution at scale, built on the Model Context Protocol (MCP).

Your AI Connections Run Through Vinkius Cloud

The world's largest managed MCP catalog

Vinkius is the cloud infrastructure where AI agents connect to the software your business already runs. We handle the hosting, the security, the credentials, the uptime — you get agents that actually do things.

We operate the world's largest managed MCP catalog. Major SaaS platforms, CRMs, databases, and cloud providers — running, monitored, production-ready. This MCP server is hosted and maintained by the Vinkius Cloud for AI Agents.

> *The agent doesn't manage credentials, doesn't manage uptime, doesn't manage security. Vinkius does.*
>
> — Architecture principle

Four Pillars of the Vinkius Runtime

01 — Security by design

Credentials stay encrypted at rest via AES-256. The AI agent never touches raw keys — they're injected into a sandboxed V8 isolate at runtime. Actions are logged, and connections have an emergency kill switch.

02 — Built on MCP Fusion

This MCP server was built with **MCP Fusion**, the open-source framework (Apache 2.0) that powers the entire Vinkius catalog. Schema-as-firewall strips undeclared fields, compiled PII redaction runs at zero overhead, and cryptographic lockfiles produce git-diffable audit trails.

03 — Deterministic observability

Eight immutable metrics per endpoint: request volume, p95 latency, error rate, active connections, cost attribution. A live payload feed logs every tool call with mutation detection.

04 — Autonomous operations

Servers are deployed, monitored, and patched autonomously. New capabilities and security patches ship weekly. Zero-downtime deployments ensure continuous availability across all managed MCP servers.

AES-256	Ed25519	24h TTL	V8 Isolate
Encryption at rest	PKI vault signatures	Ephemeral session keys	Sandboxed execution

One Token. Instant Access.

Every MCP server on Vinkius is accessed through a **Connection Token**. Tokens are generated in the cloud dashboard and produce a unique MCP endpoint URL. Paste this URL into any MCP-compatible client — no SDK required.

A single token can serve **multiple AI clients simultaneously**, or you can issue separate tokens per client for granular access control. Each token tracks its own request count, last activity timestamp, and can be individually enabled or revoked.

MCP ENDPOINT

```
https://edge.vinkius.com/{token}/mcp
```

Claude · Cursor · VS Code · Windsurf · Grok · Gemini

Security Is the Architecture

Security in Vinkius is not a feature — it's the foundation of the runtime. The gateway enforces multiple independent protection layers between AI agents and third-party APIs.

01 — Ed25519 PKI Vault

Every workspace has an Ed25519 Master Key. Session keys are generated ephemerally (24h TTL) and signed by the Master Key. Credentials never leave the vault boundary.

02 — V8 Isolate Sandboxing

Tool code runs inside isolated-vm V8 isolates with 64 MB memory caps and per-request timeouts. No filesystem access, no network access except through the SSRF-guarded fetch bridge.

03 — SSRF Guard

All outbound HTTP requests are DNS-resolved and validated before execution. Private IP ranges (10.x, 172.16-31.x, 192.168.x, AWS metadata 169.254.x) are blocked at the network layer.

04 — DLP & PII Redaction

A ResponseGuard pipeline intercepts every tool response. Configurable redaction patterns strip sensitive fields (emails, SSNs, card numbers) before data reaches the AI agent.

05 — Cryptographic Audit Trail

Every request is signed into a SHA-256 hash chain with Ed25519 signatures. Events form a tamper-proof, SIEM-exportable forensic record.

06 — Honeytoken Trap System

Phantom credentials are injected into isolated environments. If a honeytoken is used outside Vinkius infrastructure, the server is quarantined instantly.

Emergency Kill Switch

EU AI Act Art. 14(1) Compliant

The kill switch is an **emergency halt** mechanism — not a simple toggle. When triggered, it executes three actions atomically:

01 — Server deactivated

The MCP server is immediately taken offline across the entire cluster.

02 — All tokens revoked

Every connection token is invalidated. Total lockout — reconnection blocked until new tokens are issued.

03 — WebSocket connections killed

Active connections terminated via Redis pubsub broadcast. Propagates to every runtime node in the cluster.

Full Visibility. Zero Guesswork.

The Vinkius cloud dashboard includes a full MCP Governance suite — real-time analytics and security controls for production AI operations.

Control Plane

KPI dashboard with request volume, latency, success rate, token consumption, and AI-generated operational briefings.

Agent Activity

Which AI clients are connecting, how often, and what they're doing — real-time session tracking.

FinOps

Cost tracking per tool, payload compression savings, budget optimization signals, and consumption trends.

Tool Health

Slowest and most error-prone tools, with actionable root-cause insights and performance baselines.

Firewall & DLP

PII redaction activity, sensitive data protection counters, and security event timeline.

Incident Log

Error trends, failure rates, status-code breakdowns, and forensic audit trail access.

> Get started at **cloud.vinkius.com** — connect your AI agent in under 60 seconds.

Forgejo (Gitea Fork) MCP

4 tools available
Cloud-hosted on Vinkius

Connect your Forgejo or Gitea self-hosted Git instance to any AI agent to handle complex DevOps tasks naturally. Instead of logging into multiple dashboards, checking version numbers across different tabs, and manually generating access tokens, you just ask your agent. You can check detailed version compatibility between Forgejo and Gitea APIs, generate new Personal Access Tokens with specific read/write scopes, or kick off a full CI/CD build by dispatching workflows directly from the chat. The Vinkius catalog makes this connection simple; once connected to any compatible client, you get access to all these Git management functions. It's about keeping your entire development process in one conversation.

Core Capabilities

01 — Audit self-hosted version compatibility

You can retrieve detailed version numbers for both Forgejo and Gitea, confirming API compatibility across your infrastructure.

02 — Generate new access tokens

The agent generates Personal Access Tokens (PATs) using your basic credentials, allowing you to grant specific read or write scopes instantly.

03 — Trigger CI/CD pipelines manually

You can execute manual workflow dispatches for Forgejo Actions directly from the chat interface to start builds or deployments.

One Click on Vinkius — From Prompt to Execution

Available at **vinkius.com/mcp/forgejo-gitea-fork** — connect your AI agent in three steps.

01 Subscribe to this MCP, providing your specific Forgejo Instance URL and necessary credentials (Token or Username/Password).

02 Your AI agent connects to the instance via the Vinkius framework, authenticating access to your repositories.

03 You simply ask your agent to perform a task—like checking versions or triggering a workflow—and it handles the rest.

The bottom line is you manage complex repository and deployment tasks without ever leaving your chat interface.

Built For

DevOps Engineers who hate context switching, Software Developers needing quick API credentials, or System Administrators responsible for auditing multiple Git instances. You're the person who gets tired of clicking through five different web portals just to check if a build passed.

DevOps Engineer

Automatically triggering CI/CD pipelines and monitoring instance versions across dev, staging, and production environments.

Software Developer

Generating API tokens for local scripts or integrating third-party tools without having to navigate the web UI's user settings page.

System Administrator

Quickly auditing self-hosted Git instances and managing access credentials via natural conversation, keeping records clean.

What Changes When You Connect

01 Version checks are instant. Instead of manually navigating the UI to check compatibility, you ask your agent to run `get_forgejo_version` and `get_gitea_version` for a full audit.

02 Token management is fast. Need an API key for a local script? Use `create_token` to generate a new Personal Access Token instantly without leaving the chat window.

03 CI/CD automation happens on command. You can trigger builds or deployments by calling `dispatch_workflow`, eliminating the need to click through multiple pipeline dashboards.

04 Reduce context switching. All your core DevOps tasks—from checking versions to running pipelines—are centralized and managed via a single conversation with your agent.

05 Deep visibility into self-hosted systems. You can verify instance health and configuration using this MCP, giving you immediate answers without having to log in repeatedly.

Real-World Applications

Audit versioning across environments

A sysadmin needs to know if their staging environment's Forgejo instance is compatible with the new Gitea API standard. They ask the agent, and it executes `get_forgejo_version` and `get_gitea_version`, immediately confirming compatibility before they write a single line of code.

On-demand credential generation

A developer is building a script that needs read access to the main repo but shouldn't have write permissions. Instead of asking an admin, they use `create_token` to generate a dedicated PAT with only the 'read:repository' scope.

Testing deployment readiness

A team lead needs to ensure the latest code pushes correctly by triggering a full build. They instruct their agent to run `dispatch_workflow` for the `release.yaml`, confirming the pipeline starts without any manual clicks.

Troubleshooting environment issues

A junior developer reports that one service is failing mysteriously. The senior engineer asks the agent to verify the overall health by running version checks, quickly diagnosing if the problem is due to API incompatibility between the two Git systems.

Patterns to Avoid

Manually checking versions

✗ AVOID

Jumping between Forgejo's settings page and Gitea's documentation just to compare version numbers. This process is slow, error-prone, and requires keeping track of multiple tabs.

✓ INSTEAD

Instead, ask your agent to run `get_forgejo_version` and then follow up with a request for the Gitea compatibility data using `get_gitea_version`. It gives you both numbers in one response.

Using old credentials

✗ AVOID

Trying to manually update an API key by copying it from notes, only to find out the scope was wrong or the token expired. This wastes time and requires a second admin request.

✓ INSTEAD

Use `create_token` directly in the chat. You tell your agent exactly which scopes (read/write) you need for the new credential set.

Forgetting to start the pipeline

✗ AVOID

Assuming that a merge into main automatically starts the necessary build, only for the deployment to fail because no one remembered to manually kick off the workflow.

✓ INSTEAD

Don't assume it. Use `dispatch_workflow` and specify the exact YAML file (e.g., 'deploy.yaml') to guarantee the required CI/CD job runs.

The Right Fit

Use this MCP if your daily work involves coordinating multiple, distinct actions across a self-hosted Git stack—like generating tokens AND checking versions AND running builds. You need an AI agent to manage these operational tasks in sequence within one chat session.

Don't use this if you only need basic, single-purpose interaction (e.g., just viewing commit history or pulling code). For simple reading/writing of repository content, stick to your standard Git CLI tools. This MCP is for the meta-tasks: the governance, auditing, and automation around the repo itself.

The headache of managing self-hosted Git infrastructure

Right now, if you need to audit your system, you're bouncing between tabs. You check Forgejo for its version, then switch over to another dashboard to see what the Gitea compatibility looks like. If you also need a new API key for a testing script, that means logging in again and navigating through user settings just to generate a PAT.

With this MCP, your agent handles it all. You simply ask: 'What are my versions, and can I get a token for read access?' The system runs the checks and generates the credential without you touching a single dashboard. It keeps your entire process contained in the chat.

Generate API tokens with `create_token`

Previously, generating an access token meant logging into the web UI, finding the 'API' section, ensuring you selected the correct scope (read or write), and then copying a long string of characters. This process is slow and requires manual verification.

Now, you just ask your agent to use `create_token` with specific scopes and user details. The token appears immediately in the chat, ready for copy-pasting. It's faster, safer, and cuts out three clicks.

Forgejo (Gitea Fork) MCP: 4 Tools

These tools allow you to audit versions, generate credentials, and automate deployments across your self-hosted Git instances.

#	TOOL	DESCRIPTION
01	`create_token`	Generates a new personal access token for a specified user with defined permissions.
02	`dispatch_workflow`	Triggers a manual workflow run within Forgejo Actions.
03	`get_forgejo_version`	Retrieves detailed version information for the connected Forgejo instance.
04	`get_gitea_version`	Gets the Gitea-compatible version data from your self-hosted Git service.

See It in Action

Real prompts you can use once this MCP is connected to your AI agent through Vinkius Cloud.

U Check the Forgejo and Gitea-compatible versions of my instance.

I've checked your instance. It is running Forgejo version 7.0.1 and is compatible with Gitea API version 1.21.0.

U Generate a new API token named 'CI-Runner' with read:repository scope for user 'admin'.

I have successfully created the token 'CI-Runner' for user 'admin'. The token is: [TOKEN_VALUE]. Please save it securely.

U Trigger the 'deploy.yaml' workflow in the 'main-app' repository owned by 'dev-team'.

The workflow 'deploy.yaml' has been successfully dispatched for dev-team/main-app. You can monitor the progress in your Forgejo Actions dashboard.

Frequently Asked Questions

01 **How do I check the version compatibility using Forgejo (Gitea Fork) MCP?**

You need to use `get_forgejo_version` for your Forgejo data and then call `get_gitea_version` to get the corresponding Gitea-compatible API data. This tells you if they talk to each other.

02 **Can I trigger a workflow manually with Forgejo (Gitea Fork) MCP?**

Yes, you use `dispatch_workflow`. You just tell the agent which YAML file and repository needs the build, and it sends the manual dispatch command for you.

03 **Does create_token require specific permissions?**

Yes, the AI agent uses your basic credentials to generate tokens. Make sure the account connected has the necessary admin rights to issue new Personal Access Tokens (PATs).

04 **Is this MCP only for Forgejo instances?**

No, it's designed for both Forgejo and Gitea-compatible self-hosted Git services. You'll need to provide the correct instance URL when setting up your connection.

05 **What if my workflow fails? Can I check the version using Forgejo (Gitea Fork) MCP?**

While this MCP doesn't monitor live build logs, it can help diagnose potential causes by running `get_forgejo_version` or `get_gitea_version`. Version mismatch is often the root cause.

Go Live in 60 Seconds

Get your connection token from **cloud.vinkius.com**, then paste the endpoint URL into any MCP-compatible client.

YOUR MCP ENDPOINT

```
https://edge.vinkius.com/[TOKEN]/mcp
```

CLIENT	WHERE TO CONFIGURE
Claude AI	Profile → Customize → Connectors → "+" → Add custom connector → Paste endpoint
Cursor	Settings → Features → MCP Servers → "+ Add New MCP Server" → Type: SSE → Paste endpoint
VS Code	Ctrl/Cmd+Shift+P → "MCP: Add Server" → add `"forgejo-gitea-fork": { "url": "…" }`
Windsurf	MCP Settings → `mcp_settings.json` → Add endpoint URL
ChatGPT	Settings → Tools & plugins → Add MCP server → Paste endpoint
Gemini	Extensions → Add MCP Server → Paste endpoint URL

ASK AN AI ABOUT THIS

Let your preferred AI explain this MCP server

- Ask ChatGPT
- Ask Claude
- Ask Perplexity
- Ask Gemini
- Ask Grok

READY TO CONNECT

Forgejo (Gitea Fork) is live on Vinkius Cloud.

Get your connection token, paste it into your AI agent, and start building. No SDK. No deployment. Just results.

Start at cloud.vinkius.com →

vinkius.com · support@vinkius.com

INDEPENDENT PLATFORM DISCLAIMER

Vinkius is an independent platform and is not affiliated with, endorsed by, sponsored by, verified by, or otherwise authorized by Forgejo (Gitea Fork). All third-party trademarks, logos, and brand names are the property of their respective owners. Their use in this document is strictly for informational purposes to identify service compatibility and interoperability.

DOCUMENT INFORMATION

Generated	June 2026
MCP Server	Forgejo (Gitea Fork) MCP
Server ID	`019e389a-9bd8-737b-9625-bcc0c43a7e01`
Platform	Vinkius Cloud for AI Agents
Endpoint	`https://edge.vinkius.com/{token}/mcp`

LICENSE & USAGE

This document is generated automatically by the Vinkius PDF Engine. Content reflects the MCP server configuration at the time of generation and may change as updates are deployed. For the most current information, visit vinkius.com/mcp/forgejo-gitea-fork.